Ilyas Sirazitdinov

Report - Fibrosis analysis

Task

The task is regular binary classification, 2 data set were given, each of them contained patches of 2 classes: pathology and normal. I was solving the task only on fibrosis dataset but I guess the same approaches can be applied on infiltration data set. So my task was to try to implement a binary classifier of fibrosis disease based on 64x64 patches of lungs.

Data set analysis

dataset_analysis.ipynb

1) *Visual analysis*. If we you look at patches of both classes it is extremely hard to find the differences between them so explicit features are almost impossible to be found.
2) *Statistical analysis*. Let's look at our images closer. The patches are presented in dicom format and every pixel of the image is coded with value from 2000 to 9000 which as far as I understand represent the shades of grey. Let's build a distribution of the pixel intensity occurrences in the dataset. The X-axis represents the shade of gray, Y-axis number of findings in the dataset. The diagrams represented on the next page.

- Blue color - positive (path) class
- Orange - negative (norm) class

As it can be seen from the Fig. 1 the distributions for both classes are almost normal with little difference in 2000-3000 for positive and 7000-8000 for negative, but in fact distribution is the same and we cannot distinguish 2 separate classes from it. The same situation in the Fig. 2, where there is a distribution of average value of all pixels of image. Not surprisingly that we can not use this feature in classifier.

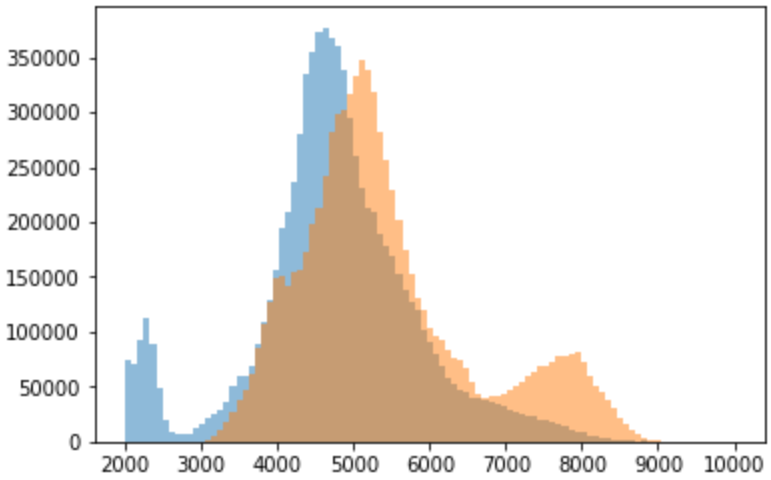

Fig.1 - Distribution of shades of grey

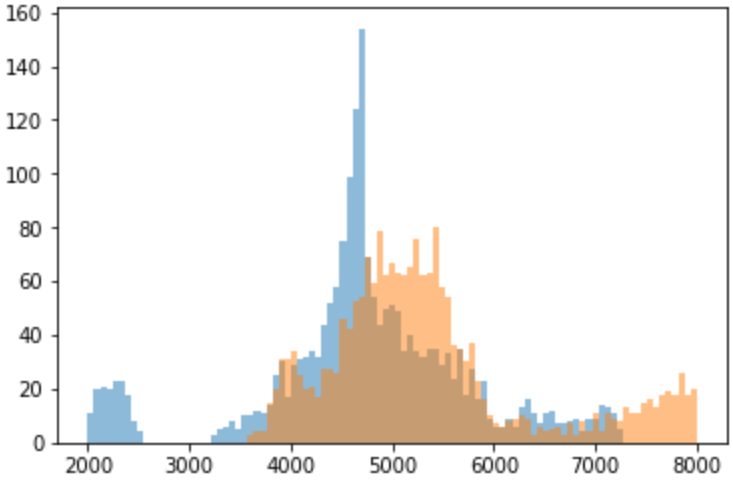

Fig.2 - Distribution of average value for every image

Method 1 - KNN or logistic regression with texture features

glcm_feat.ipynb

Obviously other features are need to be detected. So I started from a grayscale matrix of co-occurrences GLCM which is used in working with texture features. As far as the matrix has NxN dimensionality where N is a number of different colors of image it would be very

hard and ineffective to calculate 7000x7000 matrix because of its sparsity (patches are 64x64) and enormous size. So I made a preprocessing of all DICOM images and represent them as matrices 64x64 where values are in [0...255] and represent the grayscale intensity. After we calculate a GLCM matrix we can count some features based on info in it. I picked dissimilarity and correlation features but in fact others features also can be considered and may be one of combination of the features will give the best result. Let's look at the distribution of the features.

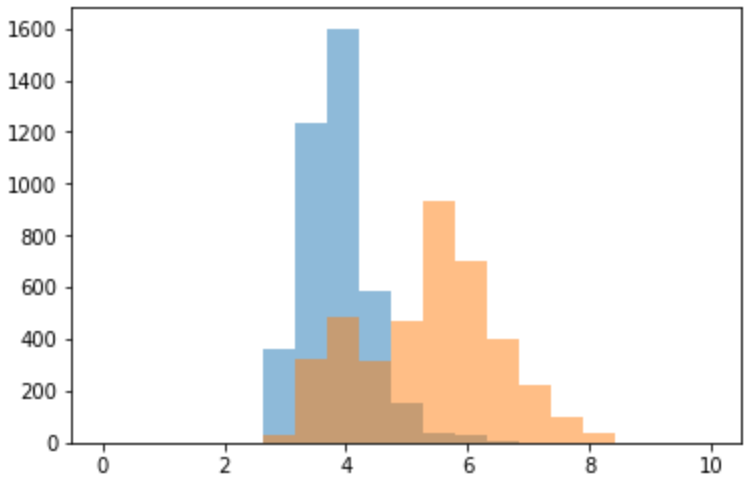

Fig.3 - Dissimilarity distribution

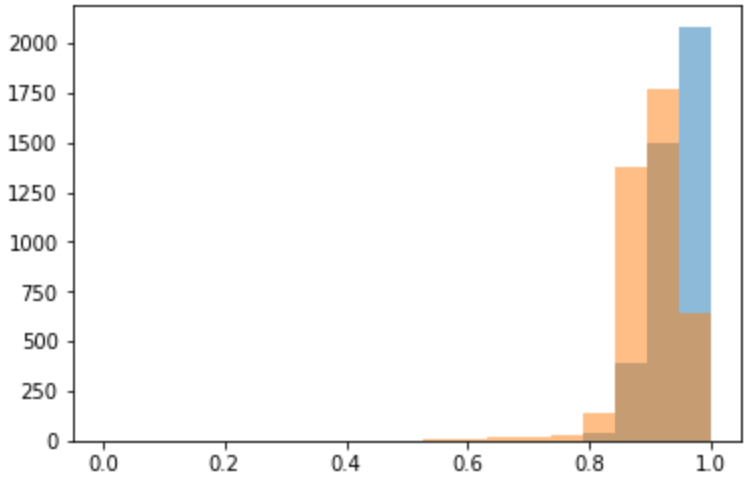

Fig.4 - Correlation distribution

As it can be noticed both of the distributions has big overlap of classes. If we represent the features in a 2-dim space we will get the picture below.

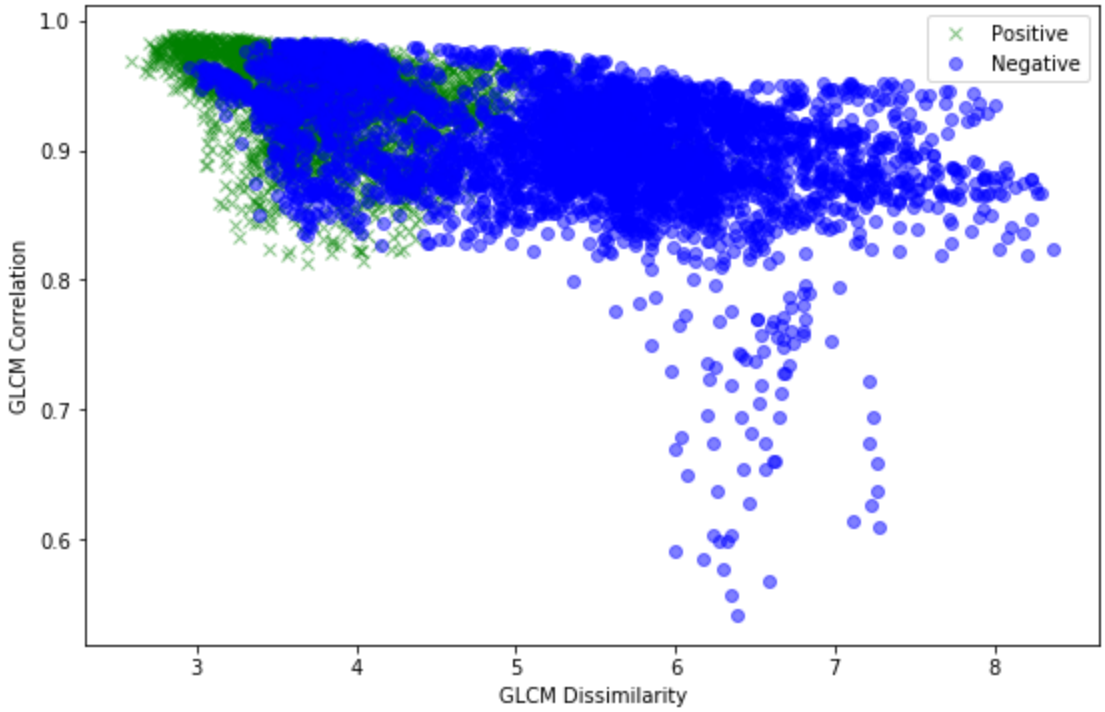

Fig.5 - 2-dim representation of features distribution

So there is still big overlap zone of positive and negative but why don't we try to implement logistic regression on this features. Obviously result won't be perfect but we will try.

So we prepare a dataset of 4000 features and validate them on 2000 testing examples. The result is: `LogisticRegression score: 0.749251`

At least it is better than random classifier. Unfortunately I'm not familiar enough with texture features and texture analysis to learn them in 2 weeks but I found a lot of interesting among them that definitely can be tested in the task.

Method 2 - Deep learning

deep_learning.ipynb

This approach based on a simple idea of allowing the neural network detect the features that I can't find. We will use all train data of fibrosis subtask which is about 20.000 patches. It is a good idea to make the augmentation of the data because the patch is small and it may be not enough to have 20.000 training set. Keras has very convenient way to perform the action, it is `flow_from_directory` which can make augmented data on the fly but the problem here is DICOM format which cannot be used in this method. It's a little time consuming task to perform augmentation of the DICOM so I will do it later and you can find a prepared code in the bottom of the notebook.

But I did deep learning on a 20.000 set. If we keep DICOM representation of image the result will be bad (I tested) so let's make a little preprocessing of the data. We normalized it in this way: new_value = (value - min_value) / (max_value - min_value) and represent as a float number.

So we have almost 20.000 training examples of 64x64 matrices which are normalized.

The structure of the network is very classical for the binary image classification and consists of convolutional layers and regular neuron layers.

Part of results of training represented below, accuracy metric was used:

```
Train on 19736 samples, validate on 4004 samples

Epoch 1/50
19736/19736 [==============================] - 81s - loss: 0.6475 -
acc: 0.6190 - val_loss: 0.5621 - val_acc: 0.7373
Epoch 2/50
19736/19736 [==============================] - 80s - loss: 0.5807 -
acc: 0.6891 - val_loss: 0.5320 - val_acc: 0.7110
Epoch 3/50
19736/19736 [==============================] - 82s - loss: 0.5337 -
acc: 0.7239 - val_loss: 0.5859 - val_acc: 0.6496
Epoch 4/50
19736/19736 [==============================] - 80s - loss: 0.4795 -
acc: 0.7614 - val_loss: 0.7152 - val_acc: 0.5937
```

As it can be seen val_acc constantly grows while training accuracy grows. It may be a problem of overfitting, so maybe we need more data. At the same time I'm not guru in convolutional nets so may be different structure of net can deal with this dataset better.

At least this approach can be used in solving the problem but I need more time to figure out some questions of neural nets and specific application with image processing.